REMARKS

Claims 1-26 are pending in this application.

Claim Rejections Under 35 USC §102

Claims 1-6, 9-15, 18-23 and 26 are rejected under 35 U.S.C. 102(b) as being anticipated by *Kush* US Patent 6,874,144 ("Kush").

Each of the pending independent claims (1, 10 and 18) have been amended to more clearly define the claims. All the independent claims call for determine whether to boost a priority of the application thread according to criteria based on future I/O operations to be performed for the application thread or whether a period of time since a last time the priority of the application thread was boosted has reached a threshold length.

Kush is concerned with situations where threads with a lower priority block threads with a higher priority. Kush does not disclose or describe a system where the decision on whether to boost a priority of the application thread is made based on the status of I/O operations performed for the application thread. Kush certainly does not contemplate a situation where future I/O operations for an application are reviewed to determined based on future I/O operations whether an application thread should receive a boost or whether any application thread has not been boosted for a given period of time.

Kush looks at the priority of the threads and attempts to create a solution to avoid having lower priority threads block higher priority threads. In fact, Kush appears to be prone to the same problem that the pending claims are attempting to address, specifically, repeated switching between I/O threads and application threads. In Kush, whether a thread is an I/O thread or any other type of thread is immaterial. All that matters is the thread priorities. As this claimed element of looking to the status of I/O operations to decide whether to boost priority of the application thread is in all the independent claims and is not present in the prior art, a prima facie case of anticipation has not been made.

In addition, the claimed solution is quite elegant and an advancement over the prior art. While Kush has to maintain multiple lists and proceed through significant thread priority inheritance reviews, the claimed system focuses on the status of I/O operations performed for the application thread. The claimed approach is smarter in that it does not automatically boost priority like Kush but intelligently determines whether two situations are occurring:

First whether just a couple of I/O operations are left (in which case application thread should not be boosted until all the I/O operations are complete);or second, if a given amount of time has passed since the application thread has been boosted (if the threshold time has passed, the priority should be boosted so that the process can pick up the waiting output before the output begins to grow too large). As a result of the claimed system, switching between I/O threads and applications threads is accomplished in a way to improve performance. The simplicity of the method reduces overhead in tracking the priority of various threads by focusing on I/O operations which limits the number of threads that need to be analyzed and tracked. This results in improved performance and is a patentable advancement over the prior art.

As this element is missing from the independent claims, it also is missing from the dependent claims. As an element is missing from all of the dependent claims, these claims should be allowed.

Claim Rejections Under 35 USC §103

Claims 7-8 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Kush* in view of *Accapadi et al.* US Publication 2005/0022186 ("Accapadi").

Dependent claims 7, 9, 15, 17, 23 and 25 have been amended to call for determining the number of I/O threads left to complete for an application process an if the number is below a threshold, refraining from raising the priority of the application thread until the I/O threads have completed. For the same reason as discussed in regard to the anticipation rejection, a prima facie case of obviousness has not been made against claims 7-8. Both Kush and Accapadi fail to disclose determining whether to boost a priority of the application thread according to criteria **based on I/O operations to be performed for the application thread in the future** (emphasis added) as called for in all the dependent claims. Accapadi adds methods to boost certain threads when the threads are about to be blocked but it does not make a determination based on a status of I/O operations performed for the application thread.

As the element of determining whether to boost a priority of the application thread according to criteria **based on a status of I/O operations performed for the application thread in the future** (emphasis added) is missing in the independent claims, it also is missing from dependent claims 7-8 and a prima facie case of obviousness has not been made.

CONCLUSION

In view of the above amendment and arguments, the applicant submits the pending application is in condition for allowance and an early action so indicating is respectfully requested.

The Commissioner is authorized to charge any fee deficiency required by this paper, or credit any overpayment, to Deposit Account No. 13-2855, under Order No. 30835/302629, from which the undersigned is authorized to draw.

Dated: September 21, 2007

Respectfully submitted,

By____/W. J. Kramer/_____
William J. Kramer
Registration No.: 46,229
MARSHALL, GERSTEIN & BORUN LLP
233 S. Wacker Drive, Suite 6300
Sears Tower
Chicago, Illinois 60606-6357
(312) 474-6300
Attorney for Applicant